Amendments to the Drawings

The attached four sheets of drawings include changes to Figs. 9A, 10A, 10B, 11, 14C and 16, and replace the original sheets containing those figures.

In Fig. 9A, "CUM 25" has been changed to "CUM 22".

In Figs. 10A and 10B, reference numbers 1010, 1020 and 1040 have been added.

In Fig. 11, reference numbers 1120 and 1130 have been added.

In Fig. 14C, reference number 88 has been added.

In Fig. 16, blocks 1650 and 1660 have been replaced by new blocks 1650, 1660 and 1670.

Attachment: Replacement Sheets
Annotated Sheet Showing Changes

REMARKS/ARGUMENTS

The specification has been amended to provide serial numbers for the related applications identified on page 1 of the application. No new matter has been added by the amendments.

Claims 1-23 are pending in the present application. Claim 9 was amended. No claims were added or canceled. Applicants believe claims 1-23 patentably distinguish over the cited art in their present form, and respectfully request reconsideration in view of the above amendments and the following comments.

I. Drawings

The Examiner has objected to the drawings as failing to comply with 37 CFR 1.84 because of various informalities.

By the present Amendment, new drawing sheets correcting Figures 9A, 10A, 10B, 11, 14C and 16 are enclosed. The drawings are now believed to be in proper form, and the Examiner is thanked for bringing these inadvertent errors to Applicants' attention.

II. 35 U.S.C. § 101

The Examiner has rejected claims 9-16 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

In rejecting the claims, the Examiner states:

> In regard to independent claim **9**, the specification refers to a "computer readable medium" both statutory and non-statutory computer readable mediums. Specifically, the specification defines "computer readable medium" as comprising "signal bearing media" comprising transmission media (see specification, page 55, last paragraph - page 56, first paragraph). A product is a tangible physical article or object, some form of matter, which a signal is not. A signal, a form of energy, does not fall within either of the two definitions of manufacture. Thus a signal does not fall within any of the four statutory classes of 101. See Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility, Annex IV (c), (signed 26, October, 2005) - OG Cite: 1300 OG 142. Retrieve on <http://www.upsto.gov/webloffices/com/sol/og/2005/week47/patgupa.htm>.
>
> Additionally, a program product with recordable medium is not necessary yet to be a computer readable medium <u>and</u> recorded/stored with executable instructions.
>
> Accordingly, claims **10-16** are rejected for not further limiting to cure the deficiencies addressed above in the rejected base claim. Appropriate correction is required.

Office Action dated June 18, 2007, pages 4-5.

In order to expedite prosecution, independent claim 9 has been amended to recite a computer program product that comprises a recordable-type computer readable medium having computer readable program code. This terminology is supported in the present specification, for example, on page 55, lines 26-28 and excludes transmission-type media. Claim 9 and claims 10-16 dependent thereon fully satisfy the requirements of 35 U.S.C. § 101 in all respects, and the rejection of the claims under 35 U.S.C. § 101 has been overcome.

III. <u>35 U.S.C. § 103, Obviousness (claims 1, 4, 7, 9, 12, 15, 17, 20 and 23)</u>

The Examiner has rejected claims 1, 4, 7, 9, 12, 15, 17, 20 and 23 under 35 U.S.C. § 103(a) as being unpatentable over Cohen et al., US 6,011,918 (hereinafter "Cohen"), in view of Kazi et al., "JaViz: A client/server Java profiling tool" (hereinafter "Kazi"). This rejection is respectfully traversed.

In rejecting claim 1, the Examiner states:

> In regard to claim **1, Cohen** discloses:
>
> > *"A method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program ..."* (E.g., see Figure 5 & Column 12, lines 25-28), wherein each detailed trace file is analyzed to gathered in the merge step, comprising average execution time for each method.
> > "*..obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program..* . "(E.g., see Figure 8 & Column 10, lines 46-51), wherein call graphs with weighted nodes are disclosed. - ". . "*...adding the call tree data structures to generate an added call tree data structure; calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure ...*" (E.g., see Figure 5 & Column 12, lines 25-28), wherein profiles are collected for multiple runs and the results associated with each node of the call tree are averaged. It is also noted that the generated added call tree data structure, although not expressly disclosed or displayed, must be created in order to compute the average. The averaging of a metric for a respective node, is in a hierarchical structure, even if in intermediate textual format, and necessarily added and divided to obtain the average of the particular metric.
>
> But **Cohen** does not expressly disclose *"...the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure"*. However, **Kazi** discloses:
> "*. . . and outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.*" (E.g., see Table 1 & page 100, "Run-time statistics generation"), wherein to facilitate the performance analysis of the call graph, statistical information is averaged (execution times) and the standard deviation, for each method, thereby minimizing the display for each execution.

Cohen and **Kazi** are analogous art because they are both concerned with the same field of endeavor, namely, a distributed application profiling tool. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine **Kazi's** averaging minimization with **Cohen's** profiling method. The motivation to do so would have been to discover the amount of time spent in cert6ain methods as disclosed by Kazi (See page 96, "Inefficient methods") to analyze the performance of the java application program.

Office Action dated June 18, 2007, pages 6-7.

Claim 1 is as follows:

1. A method, in a data processing system, for averaging out variations in trace data obtained from a plurality of executions of a computer program, comprising:
obtaining call tree data structures corresponding to the trace data for the plurality of executions of the computer program;
adding the call tree data structures to generate an added call tree data structure;
calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure; and
outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka,* 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974). In this case, the Examiner has not met this burden because all of the recited features of the claims are not found in the cited prior art references as believed by the Examiner. In particular, Neither Cohen nor Kazi nor their combination discloses or suggests the claimed steps of "adding the call tree data structures to generate an added call tree data structure", "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure", or "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure".

Cohen is directed to a mechanism for generating client/server applications from an application written to execute on a single processing system. Cohen illustrates a call graph in Figure 4 wherein nodes of the call graph represent classes of an application and subnodes correspond to programmed methods associated with each class. With reference to Figure 7, Cohen describes a process of assigning weights to the nodes (Fig. 6 of Cohen illustrates a call graph after weights have been assigned). The node weights are used to represent an approximation of client resource requirements such as, for example, the amount of RAM required to instantiate the object of a class (see column 10, lines 27-31 of Cohen).

The Examiner refers to column 12, lines 25-28 of Cohen as disclosing "adding the call tree data structures to generate an added call tree data structure", and "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure". Column 12, lines 18-36 is reproduced below for the convenience of the Examiner:

> Similarly, the computation of node and edge weights may be accomplished using dynamic information generated by profiling an exemplary execution of the program. In such an implementation profiling information rather than static analysis would be utilized to generated the weights. In a profiled implementation, the program is run such that it simulates a typical execution of the program, and profiling information is collected. Methods of collecting profiling information are well-known to those of skill in the art. Multiple runs can also be used, in which case, the results may be averaged. Based on the profiling information, node and edge weights may be assigned. In this case, node weights (single values or vectors) may be determined based on the actual resources consumed during the profiling run, optionally scaled by an uncertainty factor. Edge weights are assigned based on the actual number of calls made from one method to another and the volume of data passed during those calls. The remainder of the partitioning process may then be carried out as described above.

Initially, the above recitation states simply that "Multiple runs can also be used, in which case, the results may be averaged". This recitation does not describe how averaging is achieved, and certainly does not describe a method for averaging variations in trace data that includes "adding the call tree data structures to generate an added call tree data structure" and "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure."

Furthermore, Cohen does not disclose that the averaging referred to above is with respect to call tree data structures. Claim 1 recites that call tree data structures corresponding to trace data for a plurality of executions of the computer program are obtained, and that the adding and calculating steps are with respect to the call tree data structures. Cohen appears to disclose only that profiling information is averaged, and that "Based on the profiling information, node and edge weights may be assigned". There is nothing in Cohen to suggest that call tree data structures are added to generate an added call tree data structure, or that an average of values associated with each node in an added call tree data structure are calculated to generate an averaged call tree data structure. Only the present application contains such a disclosure.

Therefore, Cohen does not disclose or suggest the steps of "adding the call tree data structures to generate an added call tree data structure", or "calculating an average of values associated with each node in the added call tree data structure to generate an averaged call tree data structure" as recited in claim 1.

Kazi does not supply the deficiencies in Cohen. The Examiner cites Kazi as disclosing "outputting the averaged call tree data structure, wherein the affect of variations in trace data of various executions of the computer program are minimized in the averaged call tree data structure". Applicants respectfully disagree. As noted by the Examiner, Kazi states, on page 100 thereof "To facilitate the performance analysis of the call graph, statistical information about each of the methods in the program is gathered in the merge step". The merge step is described beginning on page 99 and involves merging trace files to produce "one detailed trace for each jvm that contains all of the relevant information, including the client/server activity".

At best, accordingly, Kazi may disclose merging traces to provide a <u>detailed trace</u> to facilitate performance analysis. This is not a disclosure of outputting or otherwise providing an averaged call tree data structure, <u>wherein the affect of variations in trace data of various executions of the computer program are minimized</u> in the averaged call tree data structure". To the contrary, Kazi provides a merged trace to provide more detailed information, not to minimize the effect of variations in various executions of a computer program.

For at least all the above reasons claim 1 is not obvious over Cohen in view of Kazi and patentably distinguishes over the references in its present form.

Independent claims 9 and 17 recite similar subject matter as claim 1 and also patentably distinguish over Cohen in view of Kazi for similar reasons as discussed above with respect to claim 1.

Claims 4, 7, 12, 15, 20 and 23 depend from and further restrict one of the independent claims, and also patentably distinguish over the cited references, at least by virtue of their dependency. Furthermore, many of these claims recite additional features that are neither disclosed nor suggested by the cited art. For example, the references do not disclose or suggest the subject matter of claim 4 which recites that adding the call tree data structures to generate an added call tree data structure includes copying a first call tree data structure, and walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure. Kazi simply discloses traversing nodes to gather pertinent information. This is not the same as "walking a second call tree data structure over the first call tree data structure to generate the added call tree data structure". Claim 4, accordingly, and corresponding claims 12 and 20 patentably distinguish over the cited art in their own right as well as by virtue of their dependency.

Therefore, the rejection of claims 1, 4, 7, 9, 12, 15, 17, 20 and 23 under 35 U.S.C. § 103(a) has been overcome.

IV. <u>35 U.S.C. § 103, Obviousness (claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22)</u>

The Examiner has rejected claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 under 35 U.S.C. § 103(a) as being unpatentable over Cohen in view of Kazi and in further view of Alexander et al., "A unifying approach to performance analysis in the Java environment", (hereinafter "Alexander"). This rejection is respectfully traversed.

In rejecting the claims, the Examiner states:

> In regard to claim **2,** the rejections of base claim 1 are incorporated. Cohen and Kazi do not expressly disclose *"...inputting the trace data to an arcflow tool, wherein the arcflow tool generates the call tree data structures based on the trace data."*. However, Alexander discloses:
>
> > *"...inputting the trace data to an arcflow tool, wherein the arcflow tool generates the call tree data structures based on the trace data."* (E.g., see Figure **4** & page 125, "Building the arcflow model"), wherein the arcflow tool is disclosed.
>
> **Cohen, Kazi** and **Alexander** are analogous art because they are both concerned with the same field of endeavor, namely, a distributed application profiling tool. Therefore, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to combine **Alexander's** arcflow tool with **Cohen and Kazi's** profiling method. The motivation to do so would have been to allow users to view callstack trees graphically and cross-reference the various x-files to enhance the value of arcflow as disclosed by **Alexander** (See page 131, "Future work").
> In regard to claim **3**, the rejections of base claim **1** are incorporated. **Cohen** and **Kazi** do not expressly disclose *"...the call tree data structures are xtree data structures."*. However, **Alexander** discloses:
>
> > *"...the call tree data structures are xtree data structures."* (E.g., see Figure 4 & page 125, "The xtree report"), wherein the xtree report is disclosed.
>
> In regard to claim **5,** the rejections of base claim **4** are incorporated. **Cohen** teaches the adding of values associated with each node as disclosed above with relation to claim **1.** However, **Cohen** and **Kazi** do not expressly disclose "*... adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure."* However, **Alexander** discloses: - *"...a base value ...."* (E.g., see page 124, first paragraph), wherein the base, calls and cum values are disclosed.
> Therefore, it would have been obvious to add the base value of the second node tree data structure to a base value of a corresponding node in the first tree data structure to generate the average for each node metric as disclosed above.
> In regard to claim **6,** the rejections of base claim **4** are incorporated. But **Cohen** and **Kazi** do not expressly disclose *"...for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base. value of the node that exists in either of the first call tree data structure or the second call tree data structure."*. However, it would have been an inherent result of the averaging computation if a value only existed for one node.
> In regard to claim **8,** the rejections of base claim **1** are incorporated. But **Cohen** and **Kazi** do not expressly disclose

"... *wherein the values associated with each node include a base value, a number of calls, a cumulative value, and an absolute cumulative value.*". However, **Alexander** discloses: - "... *wherein the values associated with each node include a base value, a number of calls, a cumulative value, and an absolute cumulative value.*" (E.g., see page 124, first paragraph), wherein the base, calls and cum values are disclosed.

In regard to claims **10-11, 13-14** and **16,** this is a program in a computer readable medium version of the claimed method discussed above, in claims **2-3, 5-6** and **8,** wherein all claimed limitations have also been addressed and/or cited as set forth above.

In regard to claims **18-19** and **21-22**, this is an apparatus version of the claimed method discussed above, in claims **2-3** and **5-6,** wherein all claimed limitations have also been addressed and/or cited as set forth above.

Office Action dated June 18, 2007, pages 8-11.

Claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 depend from and further restrict one of independent claims 1, 9 and 17. Alexander does not supply the deficiencies in the principal references as discussed above. Therefore, the claims patentably distinguish over the cited art, at least by virtue of their dependency. Furthermore, many of these claims recite additional subject matter that is not disclosed or suggested by the cited art. For example, none of the cited references discloses or in any way suggests the subject matter recited in claim 5 "for each node that exists in both the first call tree data structure and the second call tree data structure, generating a node in the added call tree data structure by adding a base value of the node in the second call tree data structure to a base value of a corresponding node in the first call tree data structure" or claim 6 "wherein walking the second call tree data structure over the first call tree data structure includes: for each node that exists in only one of the first call tree data structure and the second call tree data structure, creating a node in the added call tree data structure having a base value corresponding to the base value of the node that exists in either of the first call tree data structure or the second call tree data structure".

The reference in Alexander to base and Cum values is not a teaching of the steps specifically recited in claims 5 and 6, and those claims patentably distinguish over the cited art in their own right as well as by virtue of their dependency.

Therefore, the rejection of claims 2-3, 5-6, 8, 10-11, 13-14, 16, 18-19 and 21-22 under 35 U.S.C. § 103(a) has been overcome.

V. Conclusion

For at least all the above reasons, this application is believed to be in condition for allowance. It is, accordingly, respectfully requested that the Examiner so find and issue a Notice of Allowance in due course.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: September 17, 2007

Respectfully submitted,

/Gerald H. Glanzman/
Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

(Annotated Sheet)

~~AUS920030821US1~~ 10/777,742
Alexander, III et al.
Method and Apparatus for Averaging Out Variations
in Run-to-Run Path Data of a Computer Program

```
TOTAL: 10 CPU SECONDS

Lv   RL   CALLS        %BASE        %CUM      INDENT HkKey_HkName
0    1      1           0.00      100.00      AC_test_pidtid
1    1      1           0.00      100.00      - MAIN
2    1      1          10.00       40.00      --A
3    1      2          20.00       30.00      ---B
4    1      1          10.00       10.00      ----C
2    1      1          10.00       60.00      --B
3    1      1          10.00       50.00      ---A
4    1      1          10.00       10.00      ----C
4    1      1           0.00       30.00      ----X
5    1      1          10.00       10.00      -----+E
5    1      1          10.00       10.00      -----+F
5    1      1          10.00       10.00      -----+G
```

FIG. 7

TRACE DATA FOR
EXECUTION OF FIRST BUILD
OF COMPUTER PROGRAM

```
0 pidtid    xyz
3 > A
2 > B
7 < B
1 > C
5 > D
7 < D
```

FIG. 8A

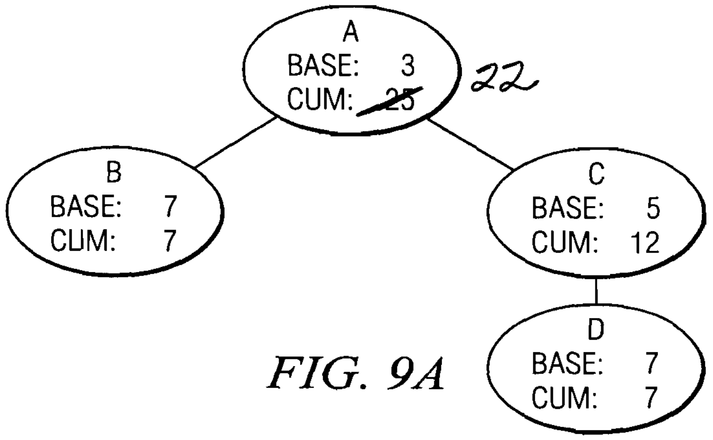


FIG. 9A

TRACE DATA FOR
EXECUTION OF SECOND BUILD
OF COMPUTER PROGRAM

```
0 pidtid    xyz
3 > A
2 > B
7 < B
1 > C
5 > E
6 < E
```

FIG. 8B

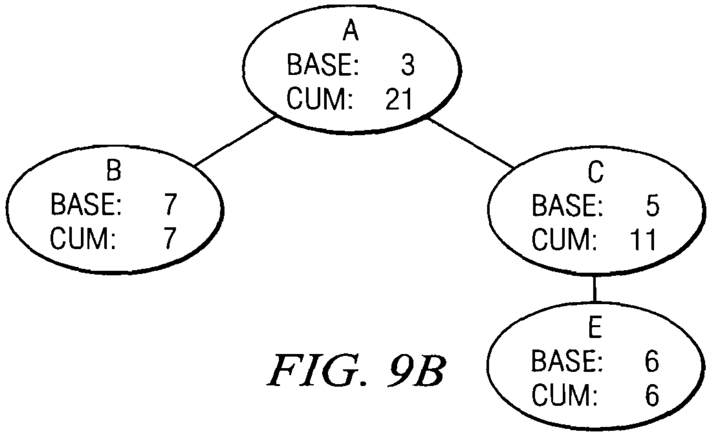


FIG. 9B

(AnNotated Sheet)

~~AUS920030821US1~~ 10/777,742

Alexander, III et al.
Method and Apparatus for Averaging Out Variations
in Run-to-Run Path Data of a Computer Program

1040 1010 1020

TOTAL: 25 CPU SECONDS

Lv	RL	CALLS	%BASE	%CUM	BASE	CUM	INDENT HkKey_HkName
0	1	2	12.00	100.00	3	25	xyz_pidtid
1	1	1	12.00	88.00	3	22	- A
2	1	1	28.00	28.00	7	7	--B
2	1	1	20.00	48.00	5	12	--C
3	1	1	28.00	28.00	7	7	---D

FIG. 10A

1030 1010 1020

TOTAL: 24 CPU SECONDS

Lv	RL	CALLS	%BASE	%CUM	BASE	CUM	INDENT HkKey_HkName
0	1	2	12.50	100.00	3	24	xyz_pidtid
1	1	1	12.50	87.50	3	21	- A
2	1	1	29.17	29.17	7	7	--B
2	1	1	20.83	45.83	5	11	--C
3	1	1	25.00	25.00	6	6	---E

FIG. 10B

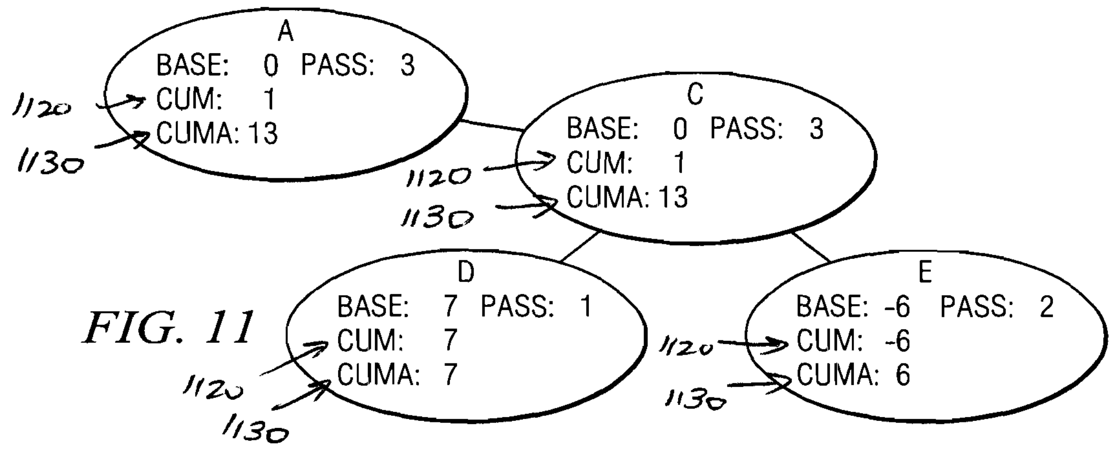


FIG. 11

TOTAL: 25 CPU SECONDS IN TREE A USED AS BASE FOR PERCENTAGES

Lv	RL	CALLS	%BASE	%CUM	BASE	CUM	CumA	PASS	INDENT HkKey_HkName
0	1	0	0.00	4.00	0	1	13		difference_pidtid
1	1	0	0.00	4.00	0	1	13	3	- A
2	1	0	0.00	4.00	0	1	13	3	--C
3	1	1	28.00	28.00	7	7	7	1	---D
3	1	-1	-24.00	-24.00	-6	-6	6	2	---E

FIG. 12

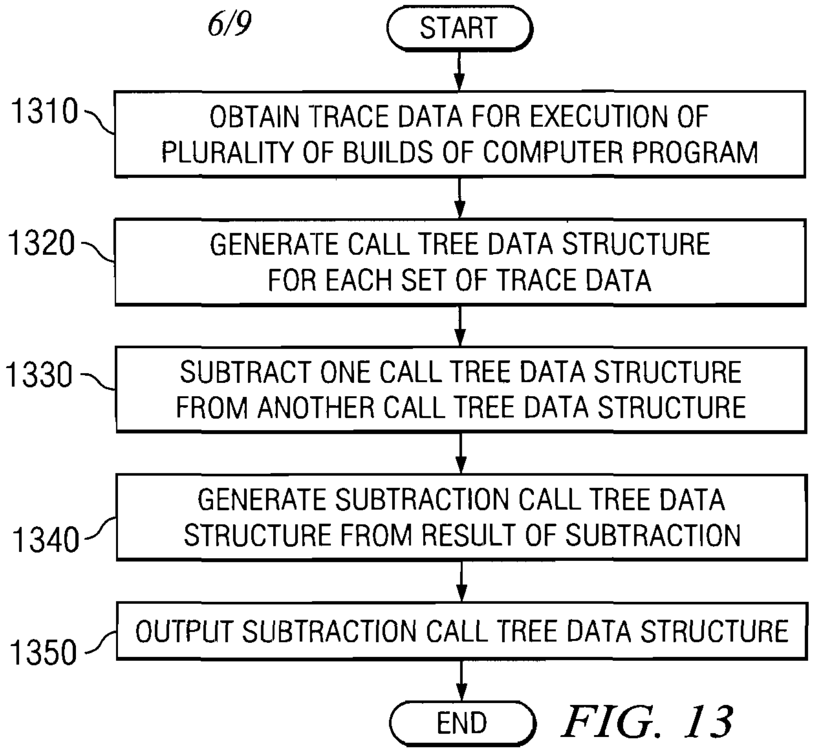

START
1310 OBTAIN TRACE DATA FOR EXECUTION OF PLURALITY OF BUILDS OF COMPUTER PROGRAM
1320 GENERATE CALL TREE DATA STRUCTURE FOR EACH SET OF TRACE DATA
1330 SUBTRACT ONE CALL TREE DATA STRUCTURE FROM ANOTHER CALL TREE DATA STRUCTURE
1340 GENERATE SUBTRACTION CALL TREE DATA STRUCTURE FROM RESULT OF SUBTRACTION
1350 OUTPUT SUBTRACTION CALL TREE DATA STRUCTURE
END

FIG. 13

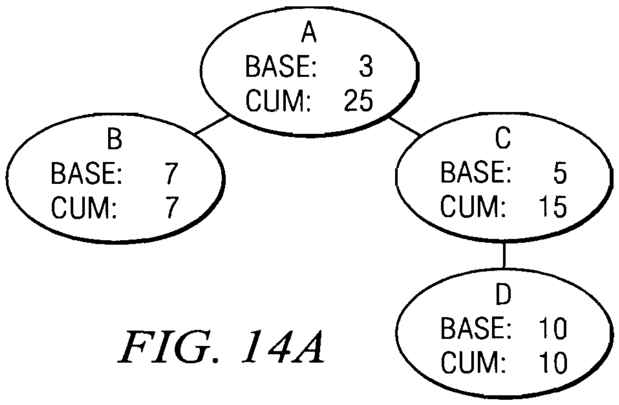


FIG. 14A

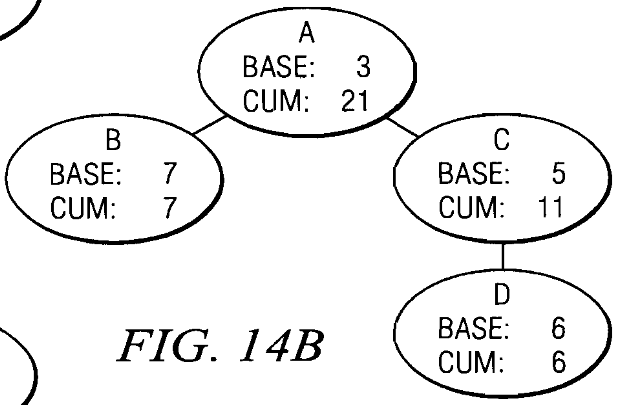


FIG. 14B

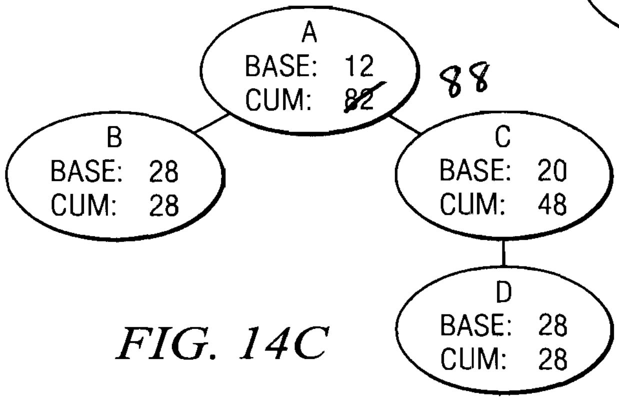


FIG. 14C

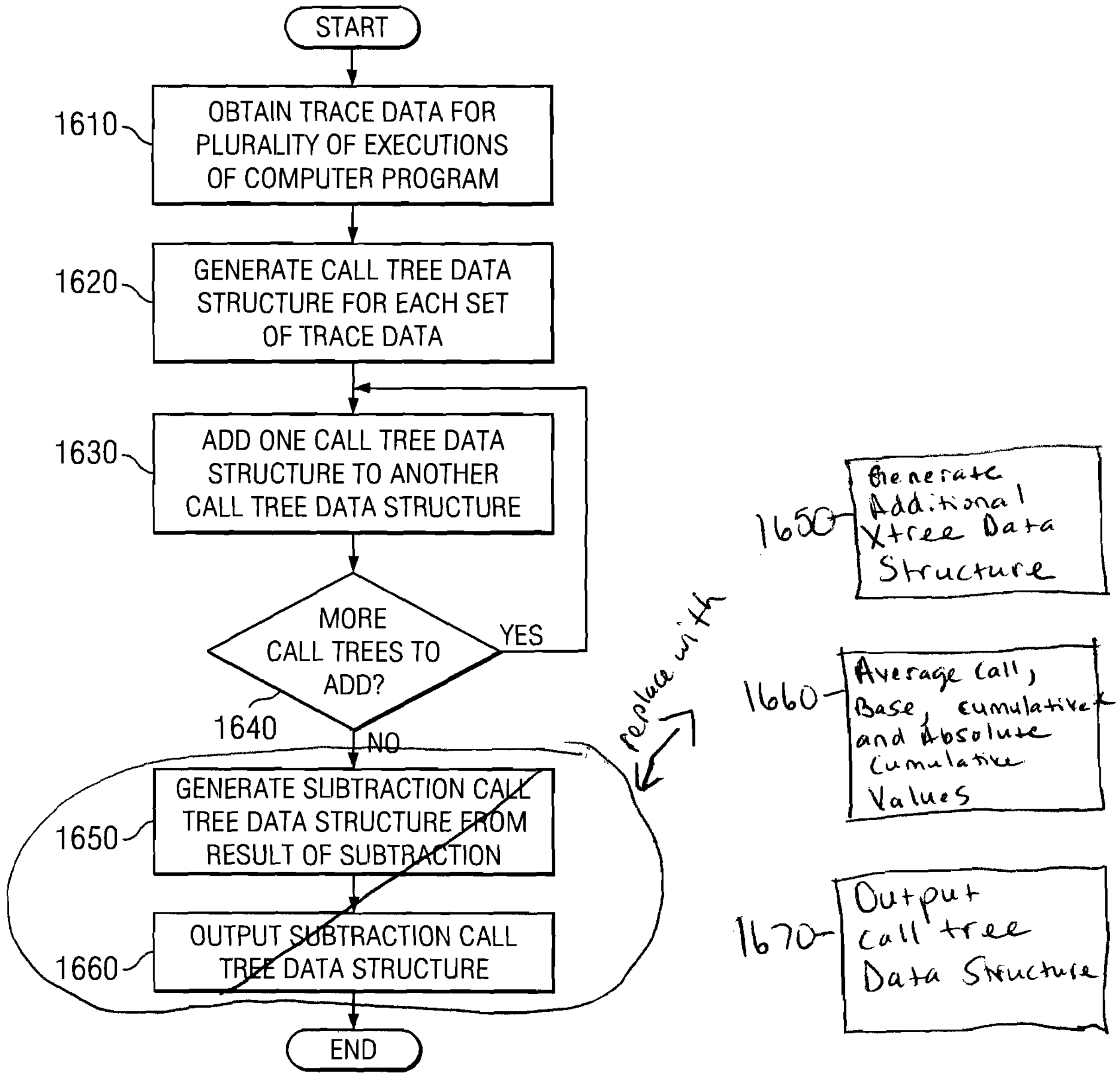
(Annotated Sheet)

~~AUS920030821US1~~ 10/777,742

Alexander, III et al.
Method and Apparatus for Averaging Out Variations
in Run-to-Run Path Data of a Computer Program

Lv	RL	CALLS	%BASE	%CUM	BASE	CUM	CumA	INDENT HkKey_HkName
0	1	3	12.16	100.00	9	74	74	bigtree_pidtid
1	1	3	12.16	87.84	9	65	65	- A
2	1	3	28.38	28.38	21	21	21	--B
2	1	3	20.27	47.30	15	35	35	--C
3	1	2	18.92	18.92	14	14	14	---D
3	1	1	8.11	8.11	6	6	6	---E

FIG. 15



START
1610
OBTAIN TRACE DATA FOR PLURALITY OF EXECUTIONS OF COMPUTER PROGRAM
1620
GENERATE CALL TREE DATA STRUCTURE FOR EACH SET OF TRACE DATA
1630
ADD ONE CALL TREE DATA STRUCTURE TO ANOTHER CALL TREE DATA STRUCTURE
MORE CALL TREES TO ADD?
YES
1640
NO
1650
GENERATE SUBTRACTION CALL TREE DATA STRUCTURE FROM RESULT OF SUBTRACTION
1660
OUTPUT SUBTRACTION CALL TREE DATA STRUCTURE
END
replace with
1650
Generate Additional Xtree Data Structure
1660
Average Call, Base, cumulative and Absolute Cumulative Values
1670
Output Call tree Data Structure

FIG. 16